PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2003-345612

(43)Date of publication of application : 05.12.2003

(51)Int.Cl. G06F 9/46

(21)Application number : 2002-154313

(22)Date of filing : 28.05.2002

(71)Applicant : SONY CORP

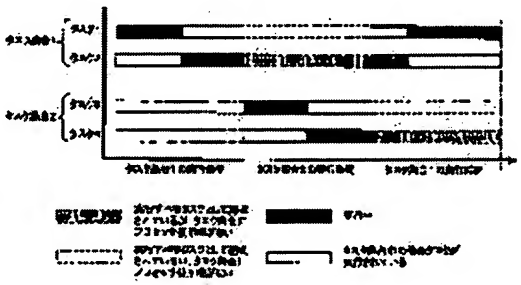
(72)Inventor : TOGAWA ATSUSHI

(54) ARITHMETIC PROCESSING SYSTEM, TASK CONTROL METHOD ON COMPUTER SYSTEM, AND COMPUTER PROGRAM

(57)Abstract:

PROBLEM TO BE SOLVED: To perform exclusive control over a critical section in the presence of a plurality of task performance environments.

SOLUTION: Provided is a mechanism which records the timing of the start of a process with high emergency and when the critical section is entered during a process with low emergency, whether a process with high emergency is to start during the performance of the critical section is examined by referring to the record. When not, the critical section is entered, but when the process is started, on the other hand, control is so performed that the entry into the critical section is prolonged until the process with high emergency is completed.



LEGAL STATUS

[Date of request for examination] 11.04.2005

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（A）　(11)特許出願公開番号
特開2003－345612
（P2003－345612A）
(43)公開日　平成15年12月5日(2003.12.5)

(51)Int.Cl.'	識別記号	FI	テーマコード'(参考)
G06F 9/46	340	G06F 9/46	340F 5B098

審査請求　未請求　請求項の数12　ＯＬ　（全 13 頁）

(21)出願番号　特願2002－154313(P2002－154313)

(22)出願日　平成14年5月28日(2002.5.28)

(71)出願人　000002185
ソニー株式会社
東京都品川区北品川６丁目７番35号
(72)発明者　戸川　敦之
東京都品川区北品川６丁目７番35号　ソニー株式会社内
(74)代理人　100093241
弁理士　宮田　正昭　（外２名）

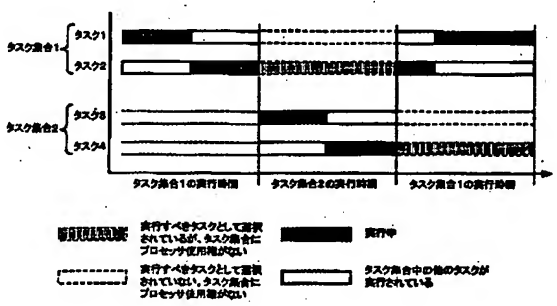
Ｆターム(参考)　5B098 AA03 GA04 GC05 GD03 GD15

(54)【発明の名称】　演算処理システム、コンピュータ・システム上でのタスク制御方法、並びにコンピュータ・プログラム

(57)【要約】

【課題】　複数のタスク実行環境が存在する下でクリティカル・セクションにおける排他制御を行なう。

【解決手段】　緊急性の高い処理が開始されているタイミングを記録する機構を備え、緊急性の低い処理の途中でクリティカル・セクションに侵入する際には記録を参照して、クリティカル・セクション実行中に緊急性の高い処理が開始されないかどうかを検査する。開始されないときにはクリティカル・セクションへと侵入し、開始される場合には、緊急性の高い処理が完了するまでクリティカル・セクションへの侵入が延期されるように制御する。



【特許請求の範囲】

【請求項１】複数の処理の間で排他制御を行なう演算処理システムであって、

緊急性の高い処理が開始されるタイミングを記録する開始タイミング記録手段と、

緊急性の低い処理の途中で複数の処理から同時に参照することができないクリティカル･セクションに進入する際に、前記の開始タイミングを参照して、該クリティカル･セクション実行中に緊急性の高い処理が開始されないかどうかを検査する検査手段と、

該検査の結果に応じて、クリティカル･セクションの実行を制御する制御手段と、を具備することを特徴とする演算処理システム。

【請求項２】前記制御手段は、該クリティカル･セクション実行中に緊急性の高い処理が開始されない場合には該クリティカル･セクションへの進入を許容し、開始される場合には該クリティカル･セクションへの進入が延期されるようにする、を具備することを特徴とする演算処理システム。

【請求項３】異なる方針に従ってスケジューリングが行なわれる複数のタスク実行環境が存在する演算処理システムであって、

タスクの実行環境を切り替える実行環境切り替え手段と、

次にタスクの実行環境の切り替えを行なう予定時刻を管理する予定時刻管理手段と、

前記予定時刻に従って現行のタスク実行環境下でのタスクの実行を管理するタスク実行管理手段と、を具備することを特徴とする演算処理システム。

【請求項４】前記タスク実行環境管理手段は、現行のタスク実行環境下で実行中のタスクが複数のタスクから同時に参照することができないクリティカル･セクションに進入する際に、前記予定時刻に対して該クリティカル･セクションの実行時間に余裕があるかどうかによって該クリティカル･セクションに進入すべきか否かを判断する、ことを特徴とする請求項３に記載の演算処理システム。

【請求項５】前記タスク実行環境管理手段は、前記予定時刻に対して該クリティカル･セクションの実行時間に余裕がある場合には該クリティカル･セクションへの進入を許容するが、余裕がない場合には前記実行環境切り替え手段に対してタスク実行環境の切り替えを指示する、ことを特徴とする請求項４に記載の演算処理システム。

【請求項６】前記実行環境切り替え手段によるタスク実行環境の切り替えに際して、切り替え時におけるタスク実行の状態を保存するコンテキスト保存手段をさらに備える、ことを特徴とする請求項３に記載の演算処理システム。

【請求項７】異なる方針に従ってスケジューリングが行なわれる複数のタスク実行環境が存在するコンピュータ・システム上でのタスク制御方法であって、

タスクの実行環境を切り替える実行環境切り替えステップと、

次にタスクの実行環境の切り替えを行なう予定時刻を管理する予定時刻管理ステップと、

前記予定時刻に従って現行のタスク実行環境下でのタスクの実行を管理するタスク実行管理ステップと、を具備することを特徴とするコンピュータ・システム上でのタスク制御方法。

【請求項８】前記タスク実行環境管理ステップでは、現行のタスク実行環境下で実行中のタスクが複数のタスクから同時に参照することができないクリティカル･セクションに進入する際に、前記予定時刻に対して該クリティカル･セクションの実行時間に余裕があるかどうかによって該クリティカル･セクションに進入すべきか否かを判断する、ことを特徴とする請求項７に記載のコンピュータ・システム上でのタスク制御方法。

【請求項９】前記タスク実行環境管理ステップでは、前記予定時刻に対して該クリティカル･セクションの実行時間に余裕がある場合には該クリティカル･セクションへの進入を許容するが、余裕がない場合には前記実行環境切り替え手段に対してタスク実行環境の切り替えを指示する、ことを特徴とする請求項８に記載のコンピュータ・システム上でのタスク制御方法。

【請求項１０】前記実行環境切り替えステップによるタスク実行環境の切り替えに際して、切り替え時におけるタスク実行の状態を保存するコンテキスト保存ステップをさらに備える、ことを特徴とする請求項７に記載のコンピュータ・システム上でのタスク制御方法。

【請求項１１】複数の処理の間で排他制御を行なうための処理をコンピュータ･システム上で実行するようにコンピュータ可読形式で記述されたコンピュータ･プログラムであって、

緊急性の高い処理が開始されるタイミングを記録する開始タイミング記録ステップと、

緊急性の低い処理の途中で複数の処理から同時に参照することができないクリティカル･セクションに進入する際に、前記の開始タイミングを参照して、該クリティカル･セクション実行中に緊急性の高い処理が開始されないかどうかを検査する検査ステップと、

該検査の結果に応じて、クリティカル･セクションの実行を制御する制御ステップと、を具備することを特徴とするコンピュータ･プログラム。

【請求項１２】異なる方針に従ってスケジューリングが行なわれる複数のタスク実行環境が存在するコンピュータ・システム上でのタスク制御の手順がコンピュータ可読形式で記述されたコンピュータ･プログラムであって、

タスクの実行環境を切り替える実行環境切り替えステッ

プと、
次にタスクの実行環境の切り替えを行なう予定時刻を管理する予定時刻管理ステップと、
前記予定時刻に従って現行のタスク実行環境下でのタスクの実行を管理するタスク実行管理ステップと、を具備することを特徴とするコンピュータ･プログラム。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】本発明は、プログラムを実行することにより所定の処理サービスを提供する演算処理システム、コンピュータ・システム上でのタスク制御方法、並びにコンピュータ･プログラムに係り、特に、プログラム中に複数の制御の流れ（例えば、割り込み処理プログラムと通常処理プログラム、又は、複数のタスクなど）が存在するタイプの演算処理システム、コンピュータ・システム上でのタスク制御方法、並びにコンピュータ･プログラムに関する。
【０００２】さらに詳しくは、本発明は、複数のタスクから同時に参照することができないプログラム部分（クリティカル･セクション）において排他制御を行なう演算処理システム、コンピュータ・システム上でのタスク制御方法、並びにコンピュータ･プログラムに係り、特に、複数のタスク実行環境が存在する下でプログラム中のクリティカル･セクションにおける排他制御を行なう演算処理システム、コンピュータ・システム上でのタスク制御方法、並びにコンピュータ･プログラムに関する。
【０００３】
【従来の技術】昨今のＬＳＩ（Large Scale Integration）技術における革新的な進歩とも相俟って、さまざまなタイプの情報処理機器や情報通信機器が開発・市販され、日常生活に深く浸透するに至っている。この種の機器では、オペレーティング･システムが提供する実行環境下で、ＣＰＵ（Central Processing Unit）やその他のプロセッサが所定のプログラム・コードを実行することによりさまざまな処理サービスを提供するようになっている。
【０００４】ところで、プログラム設計において、プログラム中に制御の流れ（「タスク」とも呼ばれる）を複数存在させることが有用な場合がある。ここで、複数の制御の流れとは、図６に示すように、プログラムの処理の流れすなわちフローチャート中に「現在実行中の地点」が複数個あることを意味する。同図に示す例では、ある時点Ｔ１では、流れＩにおいてステップＳ１が、流れIIにおいてステップＳ３が、それぞれ実行される。そして、これに対し、時間が経過して、次の時点Ｔ１では、流れＩにおいてステップＳ２が、流れIIにおいてステップＳ３が、それぞれ実行されるようになる。
【０００５】一般に、複数の流れが存在し、それぞれが共通のデータを操作する場合、これらの間で同期をとらなければデータの一貫性を保つことはできない。ここで言う共通のデータには、タスク一覧や、条件変数などが挙げられる。条件変数は、タスクが待っている条件を抽象化した概念であり、タスクがいつ待ち状態に移行すべきか、又は、いつ実行可能状態に復帰すべきかをオペレーティング・システムに伝える手段の１つとして用いられる。
【０００６】例えば、２つの制御の流れＢ及びＣが存在する場合に、それぞれの制御の流れが以下の処理を行った場合について考察してみる。
【０００７】手順１：変数ｘの値を読み出す。
手順２：読み出した値に１を加えた値を変数ｘに代入する。
【０００８】２つの流れＢ及びＣがそれぞれ上記の処理を１度行うと、同じ処理が２度行われることになる。したがって、変数ｘの値は２だけ増加するはずである。ところが、流れＢと流れＣが以下のように重なり合った場合、変数ｘの値は１しか増加しない。
【０００９】①流れＢが手順１を実行する。
②流れＣが手順１を実行する。
③流れＣが手順２を実行する。
④流れＢが手順２を実行する。
【００１０】このような動作の誤りを防ぐためには、ある流れにおいて行われた一連の参照・更新操作（トランザクション）の間に、他の流れからデータが参照・更新されることを禁じる必要がある。
【００１１】上述した例の場合、流れＢにおいて手順１と手順２という一連の操作が完全に完了する前に、流れＣが変数ｘを参照・更新してしまったために、データの一貫性が失われるという問題が生じた訳である。
【００１２】上述した手順１～２のような操作は、言い換えれば、複数のタスクから同時に参照することができないプログラム部分であり、以下では「クリティカル･セクション」とも呼ぶ。また、クリティカル･セクションにおいて、データの一貫性の問題を解決するために他のタスクによるデータの参照や更新を禁じることを「排他制御」とも呼ぶ。すなわち、ある制御の流れにおいてあるデータに対して一連の処理が行われている間は、他の制御の流れが同じデータへの操作を行うことを遅延する、すなわち特定のデータに対する操作を排他的に行なう。
【００１３】本発明者らは、排他制御機構は以下の特徴を備えているべきと思料する。
【００１４】（１）緊急度の高い処理が緊急度の低い処理によって延期される可能性（優先度逆転現象の発生）がない。
【００１５】（２）異なる方針に従ってスケジューリングが行なわれる複数のタスク集合の間でも排他制御を行なうことができる。
【００１６】上記の特徴のうち（１）が必要である理由

は当業者には明らかである。また、（2）は、1台のコンピュータ・システム上で複数のオペレーティング・システムを同時に稼動させる場合や、それぞれ特性の異なる複数のタスク集合毎に異なるスケジューリング手法を用いる場合に必要となる。

【００１７】例えば、タスク間の排他制御のために「ｍｕｔｅｘ機構」や「セマフォア機構」が使用されている。ところが、このような排他制御を用いる手法には、高優先度の処理が低優先度の処理によって延期される可能性、すなわち優先度逆転現象が発生する可能性があるという問題があるので、上記の特徴（1）を満足していない。

【００１８】このような優先度逆転の問題を緩和するための手法として、優先度継承プロトコル（例えば、Lui Sha、Ragunathan Rajkumar、及びJohn P. Lehoczky共著の論文"Priority Inheritance Protocols : An Approach to Real-Time Synchronization", IEEE Transactions on Computers, Vol. 39, No. 9, pp. 1175-1185, September 1990を参照のこと）などが提案されている。優先度継承プロトコルとは、低優先度タスクが一連の操作を実行中に高優先度タスクが同一データを操作しようとした場合には、低優先度タスクの操作が完了するまで、低優先度タスクの優先度を一時的に高優先度タスクと同一の優先度へと上昇させるという手法である。

【００１９】図7には、優先度継承プロトコルの動作を図解している。この場合、優先度の低いタスクＡがあるデータを操作中に、優先度の高いタスクＢが同じデータの操作を開始しようとしても遅延を余儀なくされる。このとき、タスクＡの優先度を一時的にタスクＢと同じレベルまで上昇させる。その後、タスクＢよりも低いがタスクＡよりも高い優先度を持つタスクＣの実行が開始しても、タスクＡの優先度はタスクＣよりも上昇しているので、タスクＡの実行が中断することはない。そして、タスクＡの終了後に、タスクＢは、データの一貫性を維持しながら、自分よりも優先度の低いタスクＣに割り込まれることなくデータの操作を開始することができる。

【００２０】ところが、この優先度継承プロトコルは、優先度という共通の尺度に従ってすべてのタスクのスケジューリングが行なわれていることを前提としている。このため、例えば単一のコンピュータ・システム上で複数のオペレーティング・システムが同時に動作するタスク実行環境のように、複数のスケジューリングが共存するシステム（特に、優先度に従ったスケジューリングを行なわないタスク集合が存在しているシステム）に対して適用することは困難である。すなわち、上記の特徴（2）を満足していない。

【００２１】これらの問題を持たない手法として、スケジューラ・カンシャス・シンクロナイゼーション手法（例えば、Leonidas I. Kontothanassis、Robert W. Wisniewski、Michael L. Scott共著の論文"Scheduler-conscious synchronization" (ACM Transactions on Computer Systems, Volume 15, Issue 1, 1997) を参照のこと）が挙げられる。この手法は、クリティカル・セクション実行中に他のタスクがディスパッチされることを禁止することによって、緊急性の高い処理に対して緊急性の低い処理が与える影響を限定している。具体的には、緊急性の高い処理の遅延時間を、最大クリティカル・セクション実行時間以下に抑えている。さらに、この手法はディスパッチを禁止する機構が備わっていることだけを前提としている。

【００２２】しかしながら、この手法も、複数のスケジューリングが共存するシステムへの適用を考慮したものではないので、このようなタスク実行環境下では、緊急性の高い処理が緊急性の低い処理によって遅延される可能性は依然として残っている。すなわち、上記の特徴（2）を満足するものではない。

【００２３】また、上記の特徴（1）及び（2）をすべて満足することができる手法として、ノンブロッキング・シンクロナイゼーション手法（例えば、Michael Barry Greenwald著の論文"Non-blocking Synchronization and System Design" (Ph. D. Thesis, Stanford University, 1999) を参照のこと）が挙げられる。しかしながら、これを適用するためには特別なハードウェアが必要となり、コスト増大を招来する。

【００２４】

【発明が解決しようとする課題】本発明の目的は、プログラム中に複数の制御の流れ（例えば、割り込み処理プログラムと通常処理プログラム、又は、複数のタスクなど）が存在するタイプの優れた演算処理システム、コンピュータ・システム上でのタスク制御方法、並びにコンピュータ・プログラムを提供することにある。

【００２５】本発明のさらなる目的は、複数のタスクから同時に参照することができないプログラム部分（クリティカル・セクション）において排他制御を好適に行なうことができる、優れた演算処理システム、コンピュータ・システム上でのタスク制御方法、並びにコンピュータ・プログラムを提供することにある。

【００２６】本発明のさらなる目的は、スケジューリング方針が異なる複数のタスク実行環境が存在する下でクリティカル・セクションにおける排他制御を好適に行なうことができる、優れた演算処理システム、コンピュータ・システム上でのタスク制御方法、並びにコンピュータ・プログラムを提供することにある。

【００２７】本発明のさらなる目的は、特別なハードウェアを必要とせずに、複数のスケジューリング方針が共存するシステムにおいて排他制御を好適に行なうことができる、優れた演算処理システム、コンピュータ・システム上でのタスク制御方法、並びにコンピュータ・プログラムを提供することにある。

【００２８】

【課題を解決するための手段及び作用】本発明は、上記課題を参酌してなされたものであり、その第１の側面は、複数の処理の間で排他制御を行なう演算処理システムであって、緊急性の高い処理が開始されるタイミングを記録する開始タイミング記録手段と、緊急性の低い処理の途中で複数の処理から同時に参照することができないクリティカル･セクションに進入する際に、前記の開始タイミングを参照して、該クリティカル･セクション実行中に緊急性の高い処理が開始されないかどうかを検査する検査手段と、該検査の結果に応じて、クリティカル･セクションの実行を制御する制御手段と、を具備することを特徴とする演算処理システムである。

【００２９】但し、ここで言う「システム」とは、複数の装置（又は特定の機能を実現する機能モジュール）が論理的に集合した物のことを言い、各装置や機能モジュールが単一の筐体内にあるか否かは特に問わない（以下同様）。

【００３０】前記制御手段は、クリティカル･セクション実行時におけるデータの一貫性を保つために、該クリティカル･セクション実行中に緊急性の高い処理が開始されない場合には該クリティカル･セクションへの進入を許容し、開始される場合には該クリティカル･セクションへの進入が延期されるようにして、排他制御を行なう。

【００３１】したがって、本発明の第１の側面に係る演算処理システムによれば、緊急性の高い処理の開始を遅らせることなく、複数の処理の間で排他制御を好適に行なうことができる。

【００３２】また、本発明の第１の側面に係る演算処理システムによれば、優先度継承を行なわないｍｕｔｅｘやセマフォと比較して、コンテンキスト切り替えの回数を削減することにより、オーバーヘッドを削減することができる。

【００３３】また、本発明の第１の側面に係る演算処理システムによれば、異なる方針に従ってスケジューリングが行なわれる複数のタスク集合の間でも排他制御を好適に行なうことができる。

【００３４】また、本発明の第１の側面に係る演算処理システムによれば、複数のオペレーティング･システムが同時に動作するシステムにおいて、これらオペレーティング･システムの上で動作しているタスク間の排他制御や、オペレーティング･システム間の排他制御が可能となる。

【００３５】また、本発明の第１の側面に係る演算処理システムによれば、特別なハードウェアを必要とせずに、複数のスケジューリング方針が共存するシステムにおいて排他制御を好適に行なうことができる

【００３６】また、本発明の第２の側面は、異なる方針に従ってスケジューリングが行なわれる複数のタスク実行環境が存在する演算処理システム又はコンピュータ・システム上でのタスク制御方法であって、タスクの実行環境を切り替える実行環境切り替え手段又はステップと、次にタスクの実行環境の切り替えを行なう予定時刻を管理する予定時刻管理手段又はステップと、前記予定時刻に従って現行のタスク実行環境下でのタスクの実行を管理するタスク実行管理手段又はステップと、を具備することを特徴とする演算処理システム又はコンピュータ・システム上でのタスク制御方法である。

【００３７】ここで、前記タスク実行環境管理手段又はステップは、現行のタスク実行環境下で実行中のタスクが複数のタスクから同時に参照することができないクリティカル･セクションに進入する際に、前記予定時刻に対して該クリティカル･セクションの実行時間に余裕があるかどうかによって該クリティカル･セクションに進入すべきか否かを判断することにより、タスクの排他制御を行なうようにすればよい。

【００３８】より具体的には、前記タスク実行環境管理手段又はステップは、前記予定時刻に対して該クリティカル･セクションの実行時間に余裕がある場合には該クリティカル･セクションへの進入を許容するが、余裕がない場合には前記実行環境切り替え手段に対してタスク実行環境の切り替えを指示するようにすればよい。

【００３９】ここで、前記実行環境切り替え手段又はステップによるタスク実行環境の切り替えに際して、切り替え時におけるタスク実行の状態を保存するコンテキスト保存手段をさらに備えていてもよい。

【００４０】本発明の第２の側面に係る演算処理システム又はコンピュータ・システム上でのタスク制御方法によれば、異なる方針に従ってスケジューリングが行なわれる複数のタスク集合の間でも排他制御を好適に行なうことができる。

【００４１】また、本発明の第２の側面に係る演算処理システム又はコンピュータ・システム上でのタスク制御方法によれば、複数のオペレーティング･システムが同時に動作するシステムにおいて、これらオペレーティング･システムの上で動作しているタスク間の排他制御や、オペレーティング･システム間の排他制御が可能となる。

【００４２】また、本発明の第２の側面に係る演算処理システム又はコンピュータ・システム上でのタスク制御方法によれば、優先度継承を行なわないｍｕｔｅｘやセマフォと比較して、コンテンキスト切り替えの回数を削減することにより、オーバーヘッドを削減することができる。

【００４３】また、本発明の第２の側面に係る演算処理システム又はコンピュータ・システム上でのタスク制御方法によれば、特別なハードウェアを必要とせずに、複数のスケジューリング方針が共存するシステムにおいて排他制御を好適に行なうことができる

【００４４】また、本発明の第３の側面は、複数の処理

の間で排他制御を行なうための処理をコンピュータ･システム上で実行するようにコンピュータ可読形式で記述されたコンピュータ･プログラムであって、緊急性の高い処理が開始されるタイミングを記録する開始タイミング記録ステップと、緊急性の低い処理の途中で複数の処理から同時に参照することができないクリティカル･セクションに進入する際に、前記の開始タイミングを参照して、該クリティカル･セクション実行中に緊急性の高い処理が開始されないかどうかを検査する検査ステップと、該検査の結果に応じて、クリティカル･セクションの実行を制御する制御ステップと、を具備することを特徴とするコンピュータ･プログラムである。

【００４５】また、本発明の第４の側面は、異なる方針に従ってスケジューリングが行なわれる複数のタスク実行環境が存在するコンピュータ・システム上でのタスク制御の手順がコンピュータ可読形式で記述されたコンピュータ･プログラムであって、タスクの実行環境を切り替える実行環境切り替えステップと、次にタスクの実行環境の切り替えを行なう予定時刻を管理する予定時刻管理ステップと、前記予定時刻に従って現行のタスク実行環境下でのタスクの実行を管理するタスク実行管理ステップと、を具備することを特徴とするコンピュータ･プログラムである。

【００４６】本発明の第３及び第４の各側面に係るコンピュータ･プログラムは、コンピュータ・システム上で所定の処理を実現するようにコンピュータ可読形式で記述されたコンピュータ・プログラムを定義したものである。換言すれば、本発明の第３及び第４の各側面に係るコンピュータ・プログラムをコンピュータ・システムにインストールすることによって、コンピュータ・システム上では協働的作用が発揮され、本発明の第１及び第２の側面に係る演算処理システム又はコンピュータ･システム上でのタスク制御方法と同様の作用効果を得ることができる。

【００４７】本発明のさらに他の目的、特徴や利点は、後述する本発明の実施形態や添付する図面に基づくより詳細な説明によって明らかになるであろう。

【００４８】

【発明の実施の形態】以下、図面を参照しながら本発明の実施形態について詳解する。

【００４９】図１には、本発明の実施に供される演算処理システム１０のハードウェア構成を模式的に示している。同図に示すように、演算処理システム１０は、プロセッサ１１と、ＲＡＭ（Random Access Memory）１２と、ＲＯＭ（Read Only Memory）１３と、複数の入出力装置１４－１，１４－２…と、タイマ１５とを含んでいる。

【００５０】プロセッサ１１は、演算処理システム１０のメイン・コントローラであり、オペレーティング・システム（ＯＳ）の制御下で、各種のプログラム・コードを実行するようになっている。

【００５１】オペレーティング・システムがプログラム実行を管理・制御する単位は、一般に「タスク」と呼ばれる。本実施形態に係る演算処理システム１０では、プログラム中に複数のタスクが存在することを許容する。したがって、実際に計算を進める実体であるプロセッサ１１の個数よりも多くのタスクが存在することになる。

【００５２】オペレーティング・システムは、プロセッサ１１により処理されるタスクを頻繁に切り替えることにより、各タスクを擬似的に並列に実行させるようになっている。各タスクには、他のタスクと識別可能なタスクＩＤが割り振られている。また、各タスクは、スタック上のデータ領域を用いてデータに対する一連の操作すなわちトランザクションを行う。

【００５３】また、本実施形態では、複数のオペレーティング･システムを同時に稼動させることができる。これらオペレーティング･システムは、それぞれ特性の異なるタスク集合を構成するとともに、異なる方針に従ってタスクのスケジューリングが行なわれる。本実施形態に係る演算処理システム１０におけるタスク実行環境の詳細については、後述に譲る。

【００５４】プロセッサ１１は、バス１６によって他の機器類（後述）と相互接続されている。システム・バス１６上の各機器にはそれぞれ固有のメモリ･アドレス又はＩ／Ｏアドレスが付与されており、プロセッサ１１はこれらアドレスを指定することによって所定の機器へのアクセスが可能となっている。システム・バス１６は、アドレス・バス、データ・バス、コントロール・バスを含む共通信号伝送路である。

【００５５】ＲＡＭ１２は、書き込み可能なメモリであり、プロセッサ１１において実行されるプログラム・コードをロードしたり、実行プログラムの作業データを一時格納するために使用される。プログラム・コードには、例えば、ＢＩＯＳ（BasicInput/Output System：基本入出力システム）、周辺機器をハードウェア操作するためのデバイス・ドライバ、オペレーティング・システム、アプリケーションなどが挙げられる。

【００５６】ＲＯＭ１３は、所定のコードやデータを恒久的に記憶するための不揮発メモリであり、例えば、ＢＩＯＳや始動時の自己診断プログラム（Power On Self Test：ＰＯＳＴ）などを格納している。

【００５７】入出力装置１４には、ディスプレイ２１を接続するためのディスプレイ・インターフェース１４－１、キーボード２２やマウス２３のようなユーザ入力装置を接続するためのユーザ入力装置インターフェース１４－２、ハード・ディスク２４やメディア・ドライブ２５などの外部記憶装置を接続するための外部記憶装置インターフェース１４－３、外部ネットワークと接続するためのネットワーク・インターフェース・カード（ＮＩＣ）１４－４などが含まれる。

【００５８】ディスプレイ・インターフェース１４－１は、プロセッサ１１が発行する描画命令を実際に処理するための専用インターフェース・コントローラである。ディスプレイ･インターフェース１４－１において処理された描画データは、例えばフレーム・バッファ（図示しない）に一旦書き込まれた後、ディスプレイ２１によって画面出力される。

【００５９】ＨＤＤ２４は、記憶担体としての磁気ディスクを固定的に搭載した外部記憶装置であり（周知）、記憶容量やデータ転送速度などの点で他の外部記憶装置よりも優れている。通常、ＨＤＤ２４には、プロセッサ１１が実行すべきオペレーティング･システムのプログラム･コードや、アプリケーション･プログラム、デバイス･ドライバなどが不揮発的に格納されている。ソフトウェア・プログラムを実行可能な状態でＨＤＤ２４上に置くことをプログラムのシステムへの「インストール」と呼ぶ。例えば、本発明を実現するオペレーティング・システムや、複数のタスクが存在するように設計されたアプリケーション・プログラムをＨＤＤ２４上にインストールすることができる。

【００６０】メディア･ドライブ２５は、ＣＤ（Compact Disc）やＭＯ（Magneto-Optical disc）、ＤＶＤ（Digital Versatile Disc）などの可搬型メディアを装填して、そのデータ記録面にアクセスするための装置である。

【００６１】可搬型メディアは、主として、ソフトウェア･プログラムやデータ･ファイルなどをコンピュータ可読形式のデータとしてバックアップすることや、これらをシステム間で移動（すなわち販売･流通･配布を含む）する目的で使用される。例えば、本発明を実現するオペレーティング・システムや、複数のタスクが存在するように設計されたアプリケーション・プログラムを、これら可搬型メディアを利用して複数の機器間で物理的に流通・配布することができる。

【００６２】ネットワーク･インターフェース１４－１は、Ｅｔｈｅｒｎｅｔ（登録商標）などの所定の通信プロトコルに従って、システム１０をＬＡＮ（Local Area Network）などの局所的ネットワーク、さらにはインターネットのような広域ネットワークに接続することができる。

【００６３】ネットワーク上では、複数のホスト端末（図示しない）がトランスペアレントな状態で接続され、分散コンピューティング環境が構築されている。ネットワーク上では、ソフトウェア・プログラムやデータ・コンテンツなどの配信が行うことができる。例えば、本発明を実現するオペレーティング・システムや、複数のタスクが存在するように設計されたアプリケーション・プログラムを、ネットワーク経由でダウンロードすることができる。

【００６４】各入出力装置１４－１，１４－２…には、割り込みレベルが割り当てられており、所定のイベント発生（例えばキーボード入力やマウス・クリックなどのＧＵＩ処理や、ハード･ディスクにおけるデータ転送の完了など）に応答して、割り込み要求信号線１９を介してプロセッサ１１に通知することができる。プロセッサ１１は、このような割り込み要求に応答して、対応する割り込みハンドラを実行する。

【００６５】タイマ１５は、タイマ信号を所定周期で発生させる装置である。タイマ１５にも割り込みレベルが割り当てられており、割り込み要求信号線１９を介してプロセッサ１１に対して周期的な割り込みを発生する。

【００６６】なお、図１に示すような演算処理システム１０の一例は、米ＩＢＭ社のパーソナル･コンピュータ"ＰＣ／ＡＴ（Personal Computer/Advanced Technology）"の互換機又は後継機である。勿論、他のアーキテクチャを備えたコンピュータを、本実施形態に係る演算処理システム１０として適用することも可能である。

【００６７】本実施形態では、複数のオペレーティング･システムが同時に稼動している。これらオペレーティング･システムは、それぞれ特性の異なるタスク集合を構成するとともに、異なる方針に従ってタスクのスケジューリングが行なわれる。

【００６８】図２には、複数のオペレーティング･システムが同時に稼動している様子を模式的に示している。同図に示す例では、２つのオペレーティング･システムＯＳ１及びＯＳ２が単一の演算処理システム１０上で同時に稼動している。ＯＳ１が提供するタスク実行環境下では、タスク１及びタスク２が実行される。また、ＯＳ２が提供するタスク実行環境下では、タスク３及びタスク４が実行される。ＯＳ１上で実行するタスク１及びタスク２はタスク集合１を構成し、ＯＳ２上で実行するタスク３及びタスク４はタスク集合２を構成する。

【００６９】演算処理システム１０上では、例えばコンテキスト切り替えなどによって、ＯＳ１とＯＳ２に対して交互にシステム使用権が与えられる。ＯＳ１にコンテキストが切り替えられた期間はタスク集合１の実行期間に相当する。そして、タスク集合１の実行期間内では、ＯＳ１が持つスケジューリングに従って、タスク集合１すなわちタスク１及びタスク２の間での排他制御が行なわれる。排他制御の手法としては、ｍｕｔｅｘやセマフォ、優先度継承プロトコル、又はその他の既存のあるいはユニークな方式を適用することができる。

【００７０】同様に、ＯＳ２にコンテキストが切り替えられた期間はタスク集合２の実行期間に相当する。そして、タスク集合２の実行期間内では、ＯＳ２が持つスケジューリングに従って、タスク集合２すなわちタスク３及びタスク４の間での排他制御が行なわれる。排他制御の手法としては、ｍｕｔｅｘやセマフォ、優先度継承プロトコル、又はその他の既存のあるいはユニークな方式を適用することができる。

【００７１】本発明は、図２に示すような、異なる方針に従ってスケジューリングされる複数のタスク集合が共存するシステムにおいて、排他制御を実現するものである。

【００７２】図３には、本発明の実施形態に係るタスク実行環境の構成を模式的に示している。同図に示すように、異なる方針に従ってスケジューリングされる複数のタスク集合が共存している。

【００７３】タスク集合１管理モジュール及びタスク集合２管理モジュールは、それぞれＯＳ１及びＯＳ２に相当し、システム使用権が与えられた期間内で所定のスケジューリング方針に従ってタスク集合１及びタスク集合２の排他制御を行なう。

【００７４】タスク集合切り替えソフトウェアは、複数のオペレーティング・システム間でのコンテキスト切り替えなどオペレーティング・システムの統括管理を行なうソフトウェア・モジュールであり、どのタスク集合から実行タスクを選出すべきかを管理することができる。タスク集合切り替えソフトウェアは、タイマ１６から時間情報を取得するタイマ・モジュールを使用して、適切なタイミングでシステムの使用権の付け替え（すなわち、タスク集合の切り替え）を行なう。

【００７５】本実施形態では、タスク集合切り替えソフトウェアは、タスク集合の切り替えを好適に管理するために、以下に示す３種類の変数を保持している。

【００７６】（１）現行タスク集合：現在プロセッサの使用権が与えられているタスク集合を識別する値が格納されている変数である。

【００７７】（２）コンテキスト保存アドレス：使用権の付け替えの際には、その時点のプロセッサの状態を保存する必要がある。変数「コンテキスト保存アドレス」は、この保存領域のアドレスを保持する変数である。

【００７８】（３）使用権移行予定時刻：次に使用権が移行する時刻を保持する変数である。この変数は、各タスク集合管理モジュールが値を読み出すことができる変数である。

【００７９】図４には、タスク集合切り替えソフトウェアがタスク集合を切り替えるための処理手順をフローチャートの形式で示している。

【００８０】タスク集合の切り替えに際し、まず、上記の変数「コンテキスト保存アドレス」に現在の状態を保存する（ステップＳ１）。ここで言う状態の保存は、より具体的には、プロセッサ１１のレジスタ値やＲＡＭ１２のメモリ・イメージの退避などを意味する。

【００８１】次いで、タスク集合の切り替えが次に行なわれる時刻を計算する（ステップＳ２）。そして、計算した時刻を上記の変数「使用権移行予定時刻」に代入する（ステップＳ３）。

【００８２】さらに、上記の変数「現行タスク集合」に、所有権移行先のタスク集合の識別子を代入する（ステップＳ４）。

【００８３】次いで、先行ステップＳ２により求められた、タスク切り替えが次に行なわれる時刻に、タイマを設定する（ステップＳ５）。

【００８４】このタイマは、設定時刻に割り込みを発生させる。この割り込みによって、次回のタスク集合切り替え処理が起動される。

【００８５】最後に、使用権の移行を、所有権移行先となるタスク集合管理モジュールに通知する（ステップＳ６）。

【００８６】図５には、あるタスク集合に属するタスクがクリティカル・セクションに侵入する際に行なう処理手続きをフローチャートの形式で示している。

【００８７】現行のタスク集合におけるタスクの実行を管理するタスク管理モジュールは、タスク集合切り替えソフトウェアが保持する変数「使用権移行予定時刻」を参照して、タスク実行の余裕を判断する（ステップＳ１１）。ここで言う余裕は、現在時刻にクリティカル・セクション実行時間を加えた値を使用権移行予定時刻から引いた値に相当する。

【００８８】次いで、この余裕がゼロよりも大きいかどうかを判断する（ステップＳ１２）。

【００８９】余裕がゼロよりも大きければ、各タスク集合に固有のクリティカル・セクションの進入処理を行なう（ステップＳ１３）。

【００９０】一方、余裕がゼロ以下であった場合には、クリティカル・セクションに突入すると、他のタスクによる同じデータへの操作が排除される一方、次のタスク集合の切り替え時刻までにクリティカル・セクションの処理が完了しないという矛盾が生じる。このため、タスク集合管理モジュールは、タスク集合切り替えソフトウェアに、プロセッサ使用権の放棄を通知する（ステップＳ１４）。

【００９１】この通知によって、処理（仮に「処理Ａ」とする）の途中に他のタスク集合中のタスクへと、制御が移ることになる。処理Ａが完了するのは、再度、処理Ａを実行していたタスクが属するタスクにプロセッサ１１の使用権が与えられたときである。

【００９２】例えば、現行のタスク集合で実行中のタスクにおいて緊急性の低い処理の途中でクリティカル・セクションに侵入する際には、次にタスク集合の切り替えが行なわれる予定時刻を参照して、クリティカル・セクション実行中に緊急性の高い処理が開始されないかどうかを検査する。そして、開始されないときにはクリティカル・セクションへと侵入するが、開始される場合には、緊急性の高い処理が完了するまでクリティカル・セクションへの侵入が延期されるように制御するようにすればよい。

【００９３】したがって、異なる方針に従ってスケジューリングが行なわれる複数のタスク集合の間で、緊急性

の高い処理の開始を遅らせることなく、複数の処理の間で排他制御を好適に行なうことができる。

【００９４】［追補］以上、特定の実施形態を参照しながら、本発明について詳解してきた。しかしながら、本発明の要旨を逸脱しない範囲で当業者が該実施形態の修正や代用を成し得ることは自明である。すなわち、例示という形態で本発明を開示してきたのであり、本明細書の記載内容を限定的に解釈するべきではない。本発明の要旨を判断するためには、冒頭に記載した特許請求の範囲の欄を参酌すべきである。

【００９５】

【発明の効果】以上詳記したように、本発明によれば、プログラム中に複数の制御の流れ（例えば、割り込み処理プログラムと通常処理プログラム、又は、複数のタスクなど）が存在するタイプの優れた演算処理システム、コンピュータ・システム上でのタスク制御方法、並びにコンピュータ･プログラムを提供することができる。

【００９６】また、本発明によれば、複数のタスクから同時に参照することができないプログラム部分（クリティカル･セクション）において排他制御を好適に行なうことができる、優れた演算処理システム、コンピュータ・システム上でのタスク制御方法、並びにコンピュータ･プログラムを提供することができる。

【００９７】また、本発明によれば、スケジューリング方針が異なる複数のタスク実行環境が存在する下でクリティカル･セクションにおける排他制御を好適に行なうことができる、優れた演算処理システム、コンピュータ・システム上でのタスク制御方法、並びにコンピュータ･プログラムを提供することができる。

【００９８】また、本発明によれば、特別なハードウェアを必要とせずに、複数のスケジューリング方針が共存するシステムにおいて排他制御を好適に行なうことができる、優れた演算処理システム、コンピュータ・システム上でのタスク制御方法、並びにコンピュータ･プログラムを提供することができる。

【００９９】本発明によれば、緊急性の高い処理の開始を遅らせることなく、複数の処理の間で排他制御を好適に行なうことができる。

【０１００】また、優先度継承を行なわないｍｕｔｅｘやセマフォと比較して、コンテンキスト切り替えの回数を削減することにより、オーバーヘッドを削減することができる。

【０１０１】また、異なる方針に従ってスケジューリングが行なわれる複数のタスク集合の間でも排他制御を好適に行なうことができる。

【０１０２】また、複数のオペレーティング･システムが同時に動作するシステムにおいて、これらオペレーティング･システムの上で動作しているタスク間の排他制御や、オペレーティング･システム間の排他制御が可能となる。

【０１０３】本発明は、ノンブロッキング･シンクロナイゼーションとは対照的に、特別なハードウェアを使用することなく、上記の効果を得ることができる。

【図面の簡単な説明】

【図１】本発明の実施に供される演算処理システム１０のハードウェア構成を模式的に示した図である。

【図２】複数のオペレーティング･システムが同時に稼動している様子を模式的に示した図である。

【図３】本発明の実施形態に係るタスク実行環境の構成を模式的に示した図である。

【図４】タスク集合切り替えソフトウェアがタスク集合を切り替えるための処理手順を示したフローチャートである。

【図５】あるタスク集合に属するタスクがクリティカル･セクションに侵入する際に行なう処理手続きを示したフローチャートである。

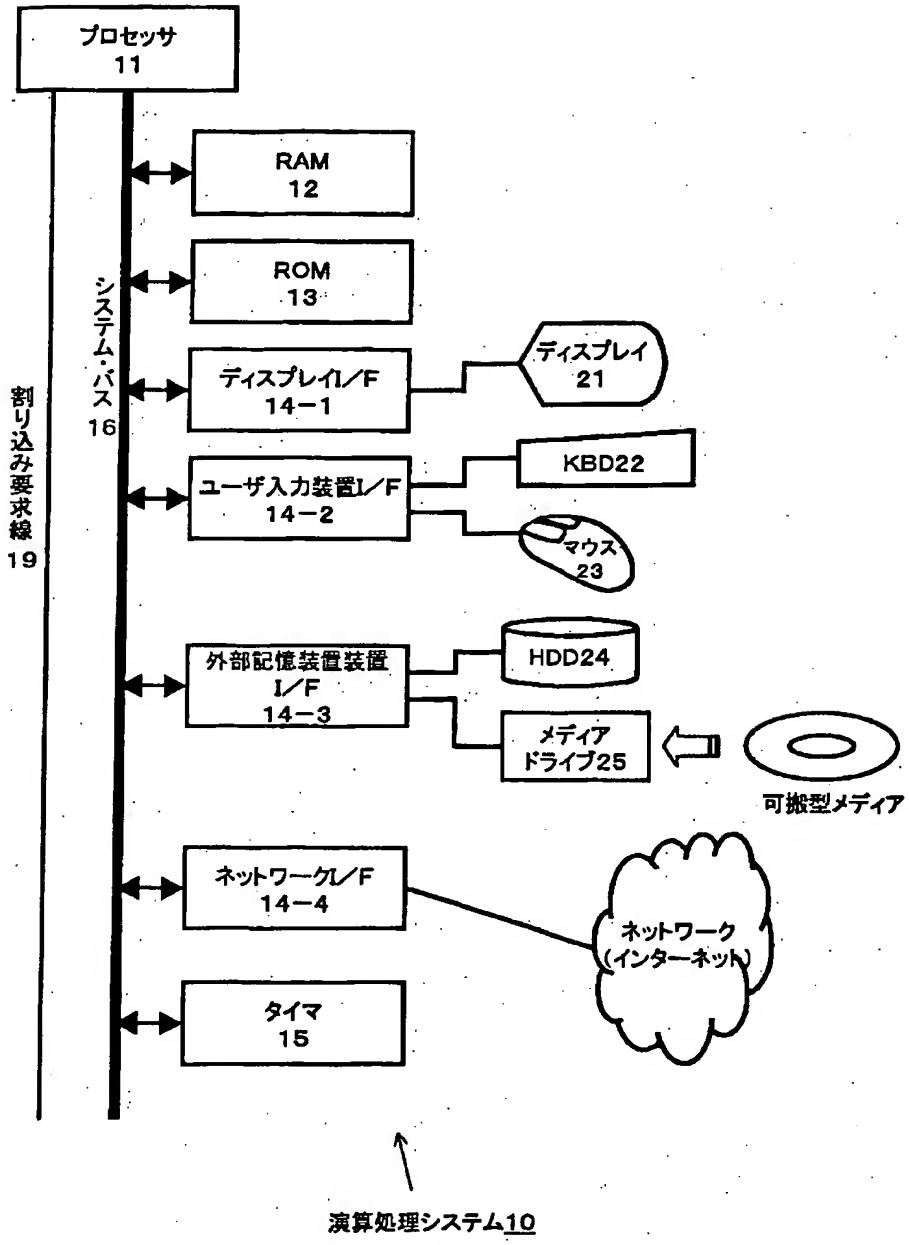
【図６】「現在実行中の地点」が複数個あるプログラムの処理の流れすなわちフローチャートを示した図である。

【図７】優先度継承プロトコルの動作を説明するための図である。

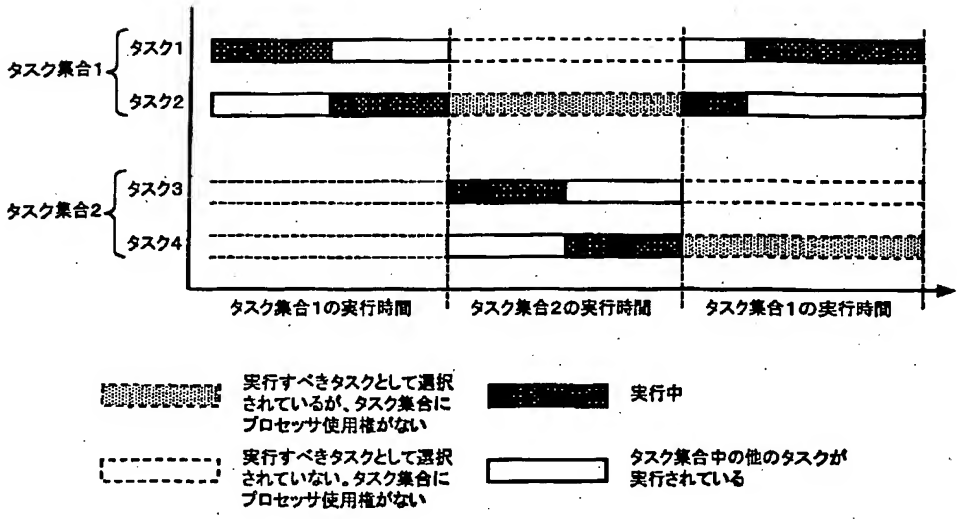
【符号の説明】

１０…演算処理システム
１１…プロセッサ
１２…ＲＡＭ
１３…ＲＯＭ
１４…入出力装置
１５…タイマ
１６…システム・バス
１９…割り込み要求線
２１…ディスプレイ
２２…キーボード
２３…マウス
２４…ＨＤＤ

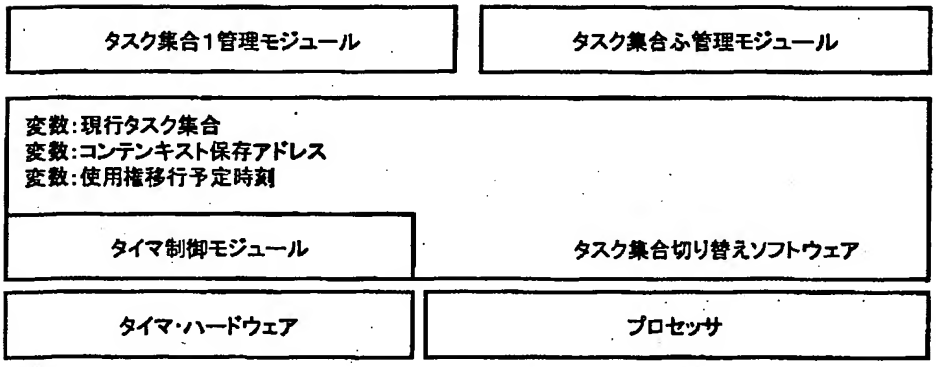
【図１】


プロセッサ
11
RAM
12
ROM
13
システム・バス
16
割り込み要求線
19
ディスプレイI／F
14－1
ディスプレイ
21
ユーザ入力装置I／F
14－2
KBD22
マウス
23
外部記憶装置装置
I／F
14－3
HDD24
メディア
ドライブ25
可搬型メディア
ネットワークI／F
14－4
ネットワーク
（インターネット）
タイマ
15
演算処理システム10

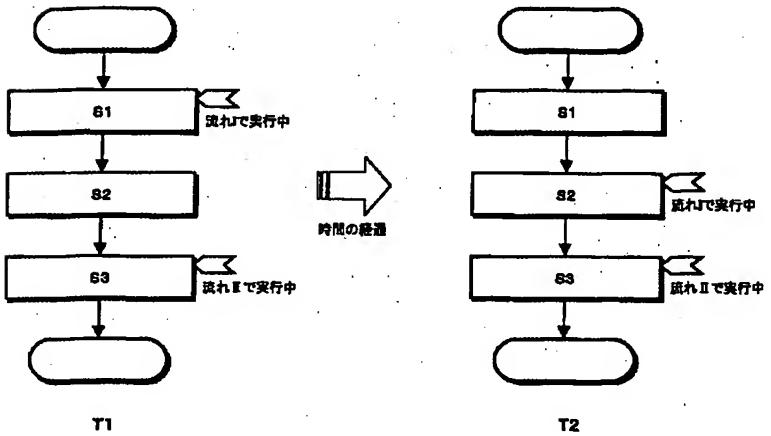
【図２】


タスク集合1
タスク1
タスク2
タスク集合2
タスク3
タスク4
タスク集合1の実行時間
タスク集合2の実行時間
タスク集合1の実行時間
実行すべきタスクとして選択されているが、タスク集合にプロセッサ使用権がない
実行中
実行すべきタスクとして選択されていない。タスク集合にプロセッサ使用権がない
タスク集合中の他のタスクが実行されている

【図３】


タスク集合1管理モジュール
タスク集合ふ管理モジュール
変数:現行タスク集合
変数:コンテンキスト保存アドレス
変数:使用権移行予定時刻
タイマ制御モジュール
タスク集合切り替えソフトウェア
タイマ・ハードウェア
プロセッサ

【図６】


S1
流れIで実行中
S2
S3
流れIIで実行中
時間の経過
S1
S2
流れIで実行中
S3
流れIIで実行中
T1
T2